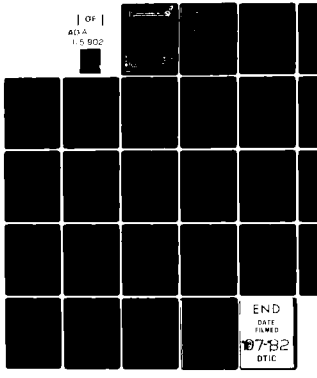
AD-A115 902 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 9/5
INFORMATION SYSTEMS DESIGN METHODOLOGY: OVERVIEW.(U)
MAY 82 D K JEFFERSON
UNCLASSIFIED DTNSRDC-82/043 NL



AD A116 [illegible]

[illegible]

[illegible] [illegible] [illegible] ENT CENTER

[illegible]

[illegible] SYSTEMS [illegible] METHODOLOGY:

[illegible]

by

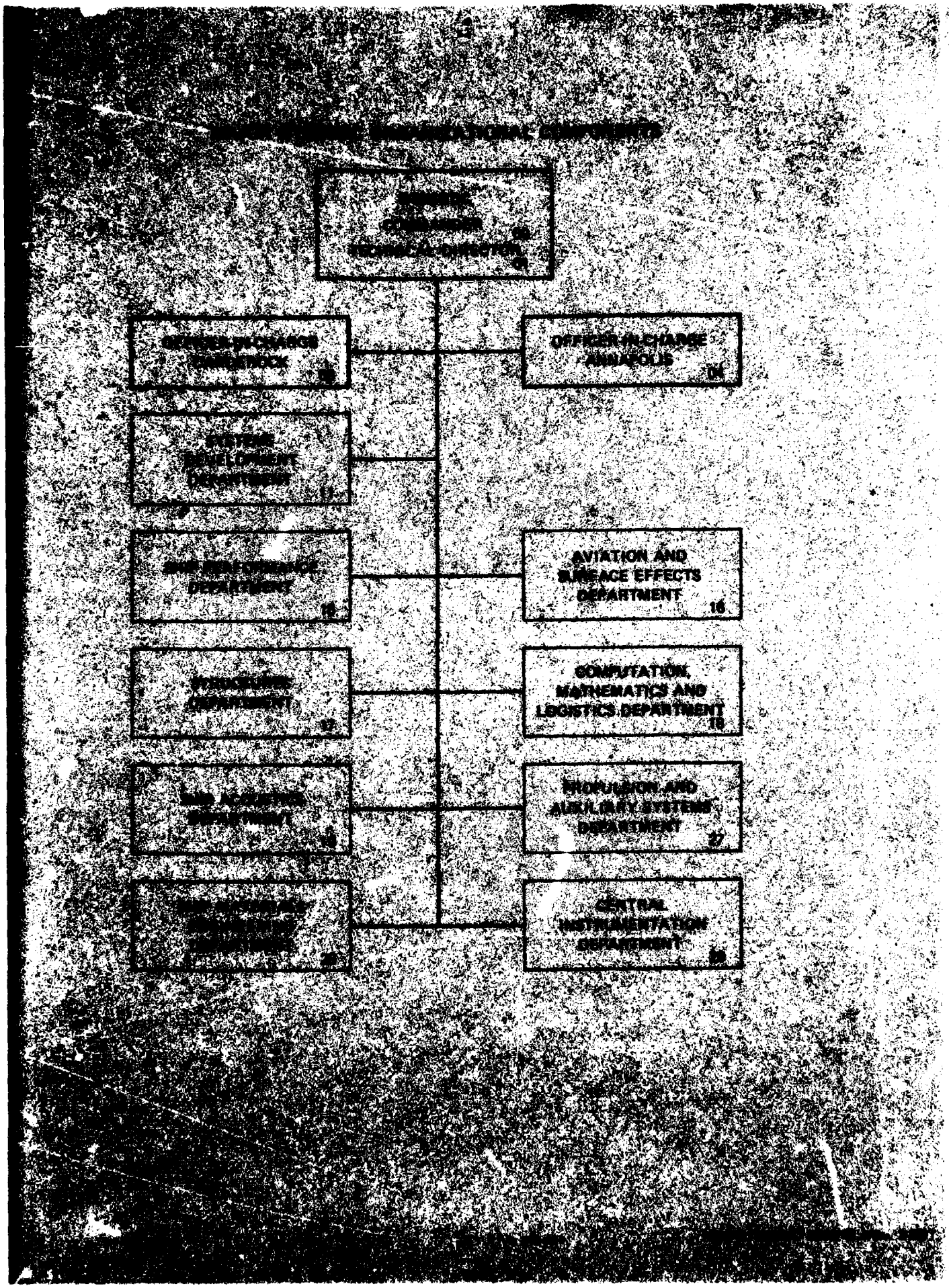
[illegible] K. Jefferson

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

[illegible] AND [illegible] DEPARTMENT

[illegible] REPORT

[illegible]


AVIATION AND
SURFACE EFFECTS
DEPARTMENT
16
COMPUTATION,
MATHEMATICS AND
LOGISTICS DEPARTMENT
18
PROPULSION AND
AUXILIARY SYSTEMS
DEPARTMENT
27
CENTRAL
INSTRUMENTATION
DEPARTMENT
28

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: DTNSRDC-82/043	2. GOVT ACCESSION NO.: AD-A115 902
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): INFORMATION SYSTEMS DESIGN METHODOLOGY: OVERVIEW	5. TYPE OF REPORT & PERIOD COVERED: Final Report June 1981-May 1982
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): David K. Jefferson	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (See reverse side)
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Supply Systems Command Research and Technology Division Washington, D.C. 20376	12. REPORT DATE: May 1982
	13. NUMBER OF PAGES: 62
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Information Systems Design
Requirements Analysis
Data Base Design
Problem Statement Language/Problem Statement Analyzer
Logistics Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This technical report briefly describes a six-phase methodology for designing an information system: formulation of the system outline, analysis of requirements, design of the global logical data base, definition of data base processes, design of the physical data base, and simulation of data base operations. The methodology is based on the extensive

(Continued on reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 10)

Program Element 62760N
Task Area TF60531100
Work Unit 1821-009

(Block 20 continued)

use of computer-aided design tools, including the Problem Statement Language/Problem Statement Analyzer (PSL/PSA). The development and application of the methodology to a very large design effort are described; numerous actual problems are described to demonstrate the need for the methodology.

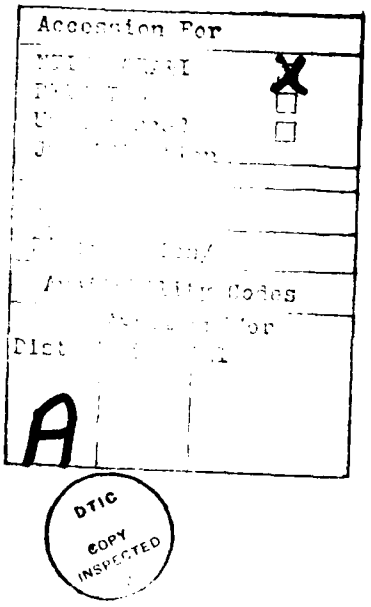


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASO - Aviation Supply Office

DEN - Data Element Number

DTNSRDC - David W. Taylor Naval Ship Research and Development Center

ECSS - Extendable Computer System Simulator

FEDSIM - Federal Computer Performance Evaluation and Simulation Center

GLDB - Global Logical Data Base

ICP - Inventory Control Point

ISDNLS - Information Systems Design for Navy Logistics Systems

ISDOS - Information Systems Design and Optimization System

NAVSUP - Naval Supply Systems Command

PSA - Problem Statement Analyzer

PSL - Problem Statement Language

RS - Requirements Statement

ABSTRACT

This technical report briefly describes a six-phase methodology for designing an information system: formulation of the system outline, analysis of requirements, design of the global logical data base, definition of data base processes, design of the physical data base, and simulation of data base operations. The methodology is based on the extensive use of computer-aided design tools, including the Problem Statement Language/Problem Statement Analyzer (PSL/PSA). The development and application of the methodology to a very large design effort are described; numerous actual problems are described to demonstrate the need for the methodology.

ADMINISTRATIVE INFORMATION

This work was performed in the Computer Sciences and Information Systems Division of the Computation, Mathematics, and Logistics Department under the sponsorship of NAVSUP 033, Task Area TF60531100, Work Unit 1821-009.

INTRODUCTION

This report describes a methodology for designing very large, complex information systems. The methodology is a research product of the Information Systems Design for Navy Logistics Systems (ISDNLS) Research Project and was tested and modified in the design of a new Inventory Control Point (ICP) System for the Naval Supply Systems Command (NAVSUP). Problems encountered in this design effort are described. Steps which should have been taken to avoid the problems, and the steps which were actually taken to compensate for the problems, are briefly discussed. Practicality is emphasized.

The remainder of this section provides a brief background on the ICP System and on the ISDNLS Research Project. The next section outlines the systems design for a loosely integrated system, in order to explain the need for additional complexity of methodology in the design of a highly integrated system. Additional sections then describe each of the phases of the methodology: what should be done, what was done, what went wrong, and what was done to make things right. A detailed technical report will be available in the future on Phase 2, Design Global Logical Data Base.

BACKGROUND ON THE ICP SYSTEM

NAVSUP's ICP System is a very large and complex system with a scope well beyond what is usually considered inventory control. The ICP System is responsible not only for some 800,000 supply items and $1.5 billion in purchases per year, but is also heavily involved in program planning, configuration management, maintenance, repair, technical documentation, and many other applications involving supply items. The current system processes 50,000 transactions a day, has about 5 billion characters of on-line storage, and is based on about 5 million lines of COBOL code and an in-house file management system. This system is now about 15 years old and its hardware limitations prohibit the development of important new applications. Maintenance of both hardware and software has become very difficult and expensive, and both hardware and software are considerably behind the state of the art. Accordingly, a new ICP System is being acquired. The new system will have at least 10 billion characters of on-line storage and will include 132 major applications. A new global logical data base has been designed, as described in "Design Global Logical Data Base." It includes about 2,000 standardized Data Element Numbers

(DEN's), about 6,000 application-oriented data elements which must eventually be standardized, 14 major data collections, 210 minor data collections, and 260 relationships among data collections. The new data base design is based on Requirements Statements (RS's) of doubtful validity, as discussed in "Analyze Requirements;" the logical design will therefore not be the basis for a physical design.

BACKGROUND ON ISDNLS

Recognition of the need for a new ICP System led to the formation in July 1974 of the Information Systems Design for Navy Logistics Systems (ISDNLS) Research Project. The objectives of this research project were to determine the state of the art in information systems design and implementation, to determine the major weaknesses of the ICP System, and to develop the technology needed to eliminate those weaknesses in a new system. Initial efforts were devoted primarily to issues of implementation and performance: data base design, distributed processing, and mass storage systems.

A fourth issue, systems analysis and design, was soon recognized as even more critical; the ICP System required complete redesign rather than isolated improvements. The system was so large and complex and had evolved over such a long period of time that modification of one part would have had tremendous impact on the entire system. Furthermore, the scope of the system was expanding far beyond the item-oriented applications for which it was originally designed. Accordingly, a major effort was devoted to systems analysis and design, in general, and to requirements analysis in particular. The Problem Statement Language/Problem Statement Analyzer (PSL/PSA)[1,2*] was acquired from the ISDOS (Information Systems Design and Optimization System) Project at the University of Michigan. PSL is a language for describing information systems; PSA is a computer program which accepts PSL statements, analyzes and stores them in a data base, and produces analyses and documentation on demand. PSL/PSA will be discussed in more detail in a later section, but the need for this tool should be immediately obvious: the requirements and designs for the new ICP System are simply too large and complex to be managed without computer assistance. The methodology described in this report is still based on PSL/PSA, which provides the complete, consistent, up-to-date requirements that are absolutely essential to

*A complete listing of references is given on page 55.

the development of a good information system. The methodology extends backward to the work which must be done before requirements analysis and forward to the work which is based on it. The following section outlines the need for the methodology.

Personnel from the ISDNLS Research Project were heavily involved from 1978 to 1982 in two aspects of the ICP System redesign: training NAVSUP personnel in the use of the information systems design methodology, and developing and maintaining the PSA data bases of design information. The problems which have been encountered, and which have caused major changes in the methodology, have generally been due much more to lack of foresight on the part of the author than to the many NAVSUP analysts who have been applying the methodology.

SYSTEMS DESIGN FOR LOOSELY INTEGRATED SYSTEMS

Systems design for loosely integrated systems, such as those based on file management systems, is accomplished by a fairly simple progression through three phases: 1) Formulation of System Concepts and Guidelines, 2) Analysis of Requirements, and 3) Detailed Design. Briefly, these phases establish why a system is needed, what the system will do, and how the system will do it. Revisions may be necessitated in the output of one phase to reflect the results of later phases (e.g., detailed design may indicate that a requirement cannot be met with the given resources), but such revisions should be rare.

FORMULATION OF SYSTEM CONCEPTS AND GUIDELINES

The result of this phase is a document describing the goals of the system and the boundaries on both development and operation. The document should address organizational goals, structure, and resources, and the system's relation to them. The system goals may include new applications, more timely production of current reports, increased security, and enhanced reliability. The description of the organizational structure should include both formal and informal lines of communication, the processes performed and their location, and the changes that can and cannot be imposed on the organizational structure to accommodate the new system. The description of available resources should include not only time, money, and personnel, but also any applicable restrictions on the amount and type of training to be required for system users, and requirements that existing hardware and software resources be used (or that they not be used). Also, any known or anticipated changes to be made in the organization's goals, structure, and resources should be specified as precisely as possible. This phase is clearly critical to the design, development, and acceptance of the system. It is also clear that top management must provide the major input to this document, must understand the document and its implications, and must ensure adherence to the document by system developers and users.

ANALYSIS OF REQUIREMENTS

The result of this phase is a document describing what the system should do, but not how it should be done. The document should include not only the functional transformations from inputs to outputs, but also timing, accuracy, reliability, and security. Subsystems and their interfaces must be defined.

FORMULATION OF DETAILED DESIGN

The result of this phase is a document describing how the requirements are to be satisfied. The document should include file and program structures and should generally address the subsystem level, since interfaces should be well-specified. (In practice, interfaces are usually specified by whatever program is implemented first.)

CRITIQUE

Subsystems may often be designed and implemented incrementally, which has the important advantage that benefits and experience are quickly acquired. Management gains confidence in the system, and problems can often be recognized before they affect much of the system. However, a loosely integrated system is likely to be quite redundant in both programs and data and will eventually become ineffective and rigid as the interfaces become more and more complex.

Systems design for highly integrated systems, such as those based on data base management systems, is ideally a much more complex and iterative process. The reason is, basically, that design now must involve the close coordination of two separate efforts: data design and process design. The availability of centrally controlled, shared data provides much more flexibility, which means more work for the designer. The phases described in the following sections show the feedback between process design and data design.

PHASE 0. FORMULATE SYSTEM OUTLINE

This phase is numbered 0, because it often has been completed before the system designers have become involved. In many cases, this phase is treated as a formal but essentially meaningless requirement, which must be completed before the "real" work of design can begin. The importance of this phase was recognized in the ICP System redesign, and the first step, formulation of system concepts and guidelines, was completed satisfactorily by a group of very experienced, high-level NAVSUP managers. However, the next two steps, defining the conceptual data structure and the functional requirements for the system, were omitted, primarily due to lack of foresight by the author, with unfortunate results in the subsequent phases. The steps that should be performed are described in the following paragraphs.

SYSTEM CONCEPTS AND GUIDELINES

This step for a highly integrated system is very much like the first phase for a loosely integrated system, but it is even more critical to the design, development, and acceptance of the system. A loosely integrated system may be designed and implemented one subsystem at a time, so that incremental benefits and experience may be quickly acquired, but a highly integrated system requires much more design work before the first subsystem can be implemented, so that both benefits and feedback are considerably delayed. The benefits of a more flexible and effective system in the long run are acquired at the cost of more planning at the beginning of system design.

CONCEPTUAL DATA STRUCTURE

The output of this step is a document describing the entities (real-world objects and concepts), and the relationships among them, that are to be modelled in the information system. The conceptual data structure provides a common language for people in different application areas. Without such a language, different names can be given to the same thing (resulting in data redundancy) and the same name can be given to different things (resulting in incorrect interfaces). The conceptual data structure also greatly facilitates later design phases by providing a framework to which more detail can be readily added.

An example of a greatly simplified conceptual data structure is shown in Figure 1. The boxes represent entities, and the lines represent relationships. A line ending in a triangle (or branching lines on other figures) indicates that many

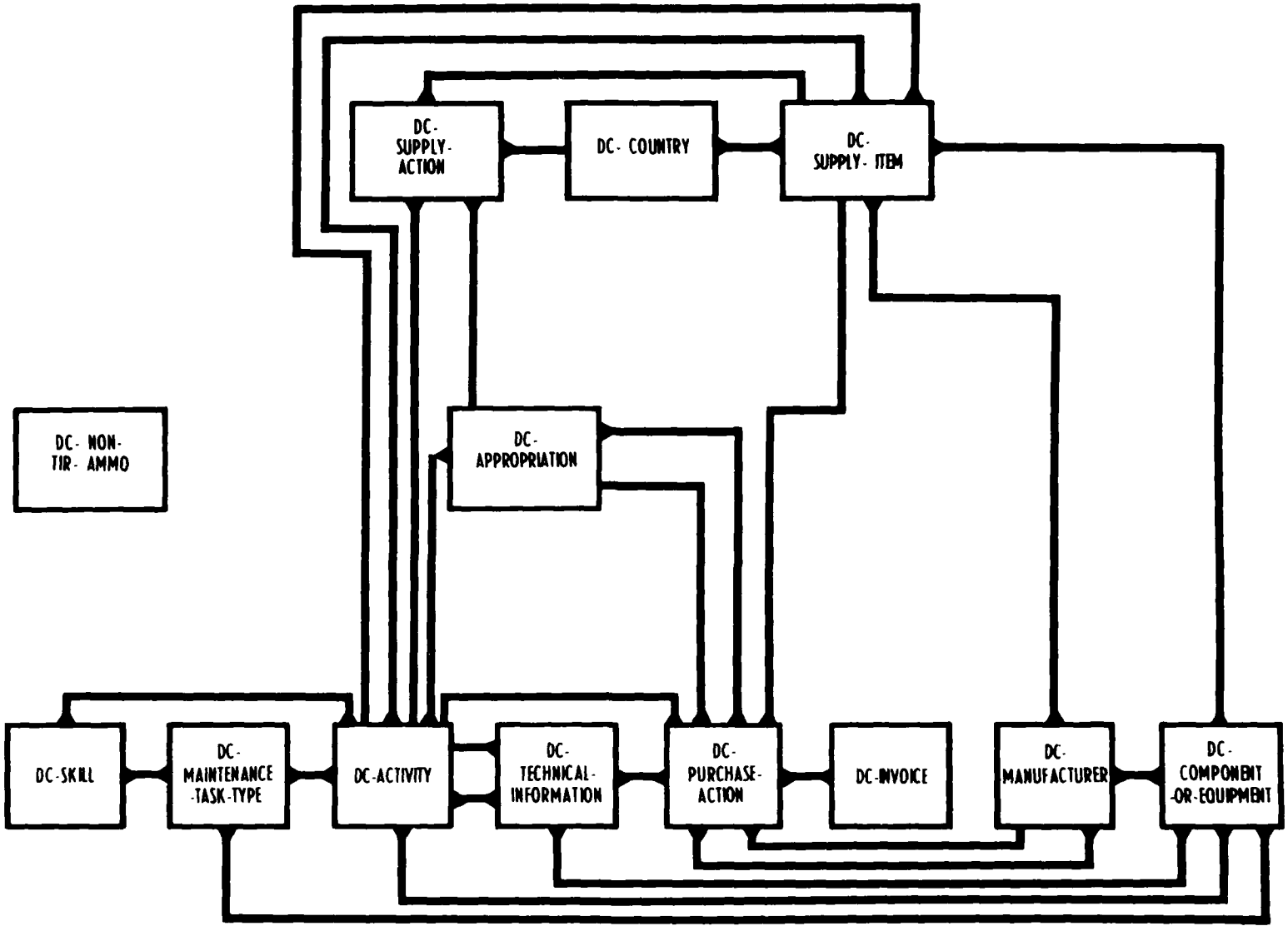


Figure 1 - Simplified Example of a Conceptual Data Structure

instances of the entity at that end of the relationship correspond to one instance of the entity at the other end. For example, a "supply item" is a real-world object that can be involved in many different "supply actions", but "supply action" is a real-world concept that can involve only a single "supply item." Lines would normally be labeled to identify the relationships.

The primary input to this step is the document on system concepts and guidelines, supplemented by interviews with middle management to determine what data are required to satisfy high-level goals.

FUNCTIONAL REQUIREMENTS FOR THE SYSTEM

The output of this step is a document describing the transformations necessary to produce the required system outputs, given the inputs. The emphasis is on what, rather than how - no attempt is made to specify algorithms, or even to decide whether the transformations are to be performed by people or by computers. The intent is to provide an outline of processing to complement and verify the conceptual data structure, to provide an initial definition of the subsystems, and to provide an initial definition of interfaces. This step will probably revise the conceptual data structure to add, combine, and refine the objects, concepts, and relationships.

The functional requirements of the system are based on system goals and the conceptual data structure, supplemented by interviews with middle management to determine what processes are required to achieve the goals. Much of this step can be done in parallel with either the preceding or following step. The development of the functional requirements for a subsystem is described in the section on analysis of requirements. The functional requirements for the system are at a much more abstract level than the functional requirements for the subsystems but are developed in the same way.

PHASE 1. ANALYZE REQUIREMENTS

The output of this phase is a document which adds a great deal of detail to the functional requirements for the system, but which is still concerned with what is to be done, rather than how it is to be done. It is likely that the preliminary definitions of the subsystems and their interfaces may be revised. However, such changes should have minimal impact on the conceptual and logical data structures, which model inherent properties of real-world objects, concepts, and relationships and are independent of the applications. The inputs for this phase consist of the logical data structure, the functional requirements of the system, and functional details provided by various people: those who will be the end users of the subsystem, management science and operations research specialists who define the algorithms, and so on.

This phase was actually performed for the ICP System redesign without the benefit of either the conceptual data structure or the functional requirements for the system; the consequences of these omissions are described in this section.

WRITE REQUIREMENTS STATEMENTS

A requirements statement (RS) was written for each of the 132 applications (subsystems). The RS's were intended to provide both the functional requirements and the logical data structures for the subsystems. The structure of an RS is shown in Figure 2. The RS's were to be combined, bottom-up, to yield the functional requirements and logical data structure for the system. This combination proved to be infeasible for the following reasons:

1. RS writers received minimal training, so that quality was often low. In particular, many did little more than write an operational specification for the existing system, making it difficult to determine what the requirements really were.

2. RS's were often very detailed and complex, were written in English with all its ambiguities, and followed a rather loose structure, so that internal completeness and consistency could generally not be checked.

3. No conceptual or logical data structure was available to the RS writers, so they devised their own data collections and even many data elements. Consequently, there was no reasonable way of combining the RS's to form a functional description of the whole system.

SAMPLE

TABLE OF CONTENTS

Figure 2 - Example of RS Structure

DRAW FUNCTIONAL DIAGRAMS

The first step toward making the RS's useful was to develop, for each RS, a hierarchy of functional diagrams (Figures 3 and 4 are examples of first and second level diagrams, respectively). These diagrams were intended to represent only the functional requirements, without the detail of the RS's. The diagramming technique, a simplified version of SADT (TM)[3], provided not only a clearer description of the functions, but also a means of checking for internal consistency and completeness. The quality of the diagrams was much higher than that of the RS's, in part because of the technique and in part because the diagrams were constructed by a much smaller, more highly trained group of people. However, the lack of consistency in data collections still made it impossible to combine the diagrams for different RS's.

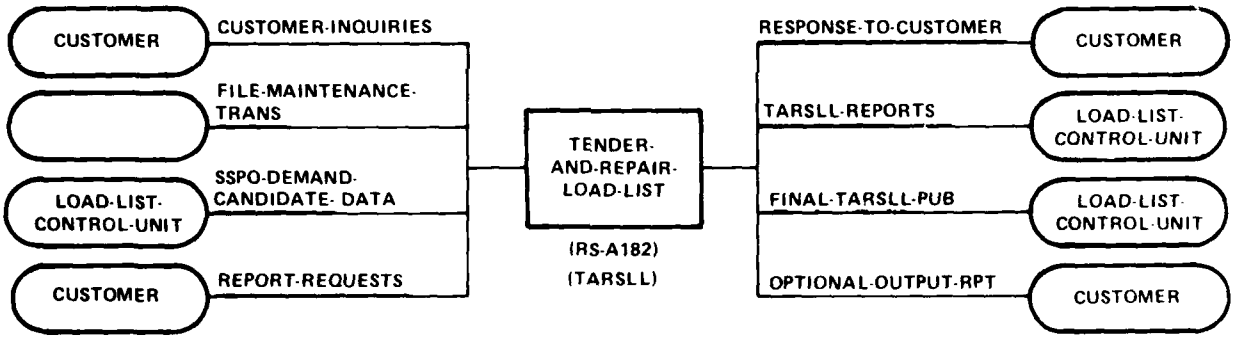


Figure 3 - Example of an Overview Diagram

MECHANIZE REQUIREMENTS STATEMENTS

The second step toward making the RS's useful was to represent them in a computer-processable form. The functional diagrams and data collections (see example in Figure 5) were expressed in the Problem Statement Language (PSL),[1,2] stored in a Problem Statement Analyzer (PSA) data base, analyzed for consistency and completeness by PSA, and documented by PSA for review by NAVSUP analysts. Mechanization of the

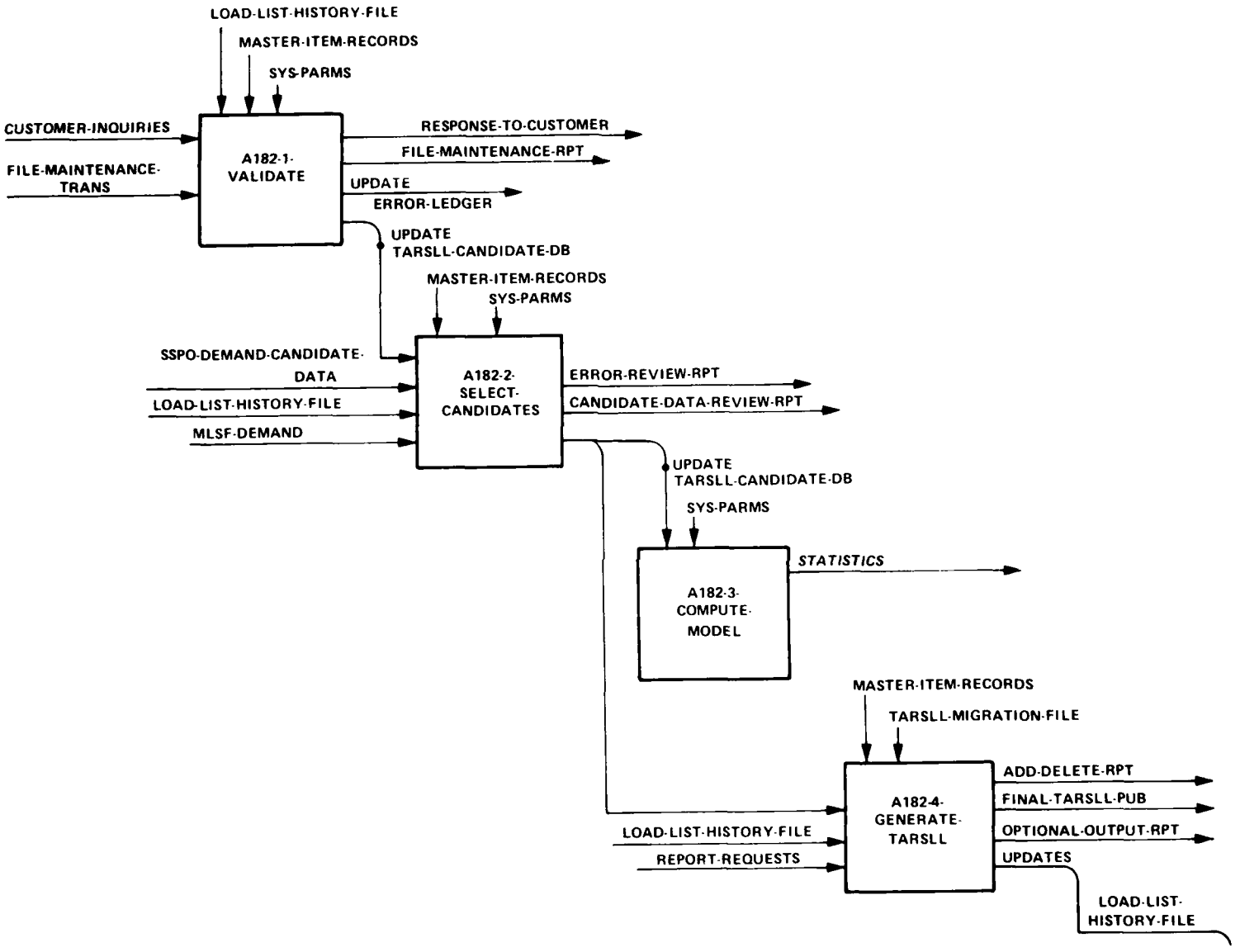


Figure 4 - Example of a Detailed Diagram

MAY 24 1979

DATA ELEMENT DEFINITION FORM

MAJOR SYSTEM NAME: UICP

SUBSYSTEM OR APPLICATION: TENDER AND REPAIR SHIP LOAD LIST

SYSTEM FUNCTION: TARSLL CANDIDATE SELECTION MODULE
MLSF DEMAND AND OVERRIDES

DEN NO.	DATA ELEMENT NAME	NO. OF CHAR	CHARACTER TYPE A=ALPHA N=NUMERIC A/N=BOTH	DATA ELEMENT DESCRIPTION	USE (✓) ACCESS ONLY	USE (✓) UPDATE
DØ46	Federal Item Identification Code (Old & New)	7	N		X	
DØ46D	National Item Identification Code (Old & New)	9	N		X	
EØØ4A	Load Activity Code	5	A/N		X	
XA18.Ø36A	Mobile Logistics Support Force Demand Frequency	4	N	MLSF Demand Frequency summarized by quarter. First Quarter = the most recent quarter. See RS A18.3.	X	
	First Quarter					
XA18.Ø36B	MLSF Demand Frequency - Second Quarter	4	N		X	
XA18.Ø36C	MLSF Demand Frequency - Third Quarter	4	N		X	
XA18.Ø36D	MLSF Demand Frequency - Fourth Quarter	4	N		X	
XA18.Ø36E	MLSF Demand Frequency - Fifth Quarter	4	N		X	
XA18.Ø36F	MLSF Demand Frequency - Sixth Quarter	4	N		X	
XA18.Ø36G	MLSF Demand Frequency - Seventh Quarter	4	N		X	
XA18.Ø36H	MLSF Demand Frequency - Eighth Quarter	4	N		X	
XA18.Ø39A	Mobile Logistics Support Force Demand Quantity -	6	N	MLSF Demand Quantity summarized by quarter. First quarter = the most recent quarter. See RS A18.3.	X	
	First Quarter					

SUBMITTED BY: | ORGANIZATION: | DATE: | PAGE NO. 1

Figure 5 - Example of a Data Collection

RS's provided a way of combining the simplicity of the functional diagrams with the element-level detail of the RS's. Mechanization also provided a reasonable way of maintaining the RS's. Finally, mechanization provided a small but useful step toward understanding the relationships among different RS's: the use of standardized Data Element Numbers (DEN's) could be readily compared across RS's. Since RS writers could also define their own non-standard data elements (those beginning with "X" in Figure 4), which generally could not be compared across RS's, RS's could be compared only to a limited degree.

Examples of the PSA reports used to check the validity of the PSA data base are shown in Figures 6-8. Figure 6 indicates that the ELEMENTs have not yet been given DESCRIPTIONs (English text carried with the data element but not analyzed by PSA). Figure 7 is a list of names; this can be visually checked for various types of consistency, such as spelling and adherence to naming conventions. Lines 51 and 52 indicate a problem, since every element should have a descriptive title as well as a number. Figure 8 indicates either that the INPUTs and OUTPUT have been incompletely defined (the sources of the INPUTs and destination of the OUTPUT are missing), or that these names are misspellings of some other data.

Figures 9-12 are examples of reports sent to the NAVSUP analysts. Figure 9 shows an overview of an RS - its INPUTs, OUTPUT, and the more detailed PROCESSes into which it is decomposed. The overview is useful primarily as a brief orientation to the RS.

Figure 10 describes a data collection (in this case a GROUP - which may be a record). Figure 11 shows briefly how the RS processing is hierarchically structured, and Figure 12 describes the interaction of each process with data.

CRITIQUE

The problems encountered in this phase could be avoided by the following procedure:

1) Develop the system outline as proposed in Phase 0. Train analysts to use that outline.
2) Develop functional diagrams to describe the functional requirements of the subsystems.
3) Store and analyze the functional requirements with a tool such as PSL/PSA.

PSA Version A5.1R5 Jun 2[illegible], 1981 7:47:49

RS-A812

Data Base Summary

Object Type	Count	Number with SYNONYM	Percent with SYNONYM	Number with DESC	Percent with DESC
ELEMENT	231	0		0	
GROUP	6	0		0	
INPUT	5	0		0	
MEMO	1	0		1	100.00
OUTPUT	1	0		0	
PROCESS	5	1	20.00	0	
** Total **	249	1	0.40	1	0.40

Figure 6 - Example of an RS Summary

4) Add additional detail to the requirements data base.

5) At all times, provide adequate training to analysts in how their requirements will be used, and limit the number of people working simultaneously so that their efforts can be closely coordinated.

The objectives of this procedure are to provide a detailed framework within which to perform each step of the design phase, to provide consistency, and to detect errors before detailed work has been done. The design steps should be accomplished, or at least closely supervised, by a small group of highly trained people.

PSA Version A5.1R5 Jun 22, 1981 10:16:52

RS-A812

Name List

	Name	Type	Synonym
1	A002-UNIT-IDENTIFICATION-CODE	ELEMENT	-
2	C001-FED-STK-NR-ACTVY-CNTRL-NR	ELEMENT	-
3	C003B-SPCL-MATL-IDFCN-CODE	ELEMENT	-
4	C008B-MEC-CMPNT-ESPMT-FBM	ELEMENT	-
5	C008D-MIL-ESENTLTY-CODE-CMPNT	ELEMENT	-
6	C011-PRVSNG-DOC-CNTRL-NR	ELEMENT	-
7	C017-SECURITY-CLASSIFICATION-C	ELEMENT	-
8	C543B-TYPE-COMMANDER-NAME	ELEMENT	-
9	D008-REPARABLE-IDENTIFICATION-	ELEMENT	-
10	D008D-EQUIPMENT-IDENTIFICATIO	ELEMENT	-
11	D009-APPLICATION-CODE	ELEMENT	-
12	D013M-MAINTENANCE-LEVEL-CAPABI	ELEMENT	-
13	D029-APLICN-IDFCN-NR-ACTVY-COD	ELEMENT	-
14	D031-LOGISTIC-SUPPORT-STATUS-C	ELEMENT	-
15	D032-SERIAL-NUMBER	ELEMENT	-
16	D034-TY-OF-TECH-DOC-CODE	ELEMENT	-
17	D036B-SHIP-TYPE-AND-HULL-NUMBE	ELEMENT	-
18	D036D-END-USE-NAME	ELEMENT	-
19	D037-DATA-CRIGR-VALDN-CODE	ELEMENT	-
20	D038-EQUIPMENT-SUPPLIERS-CODE	ELEMENT	-
21	D044-TECHNICAL-COGNIZANCE-CDE	ELEMENT	-
22	D076-INSTALL-PLAN-NUMBER	ELEMENT	-
23	D077-TYPE-OF-CHANGE-REQUEST	ELEMENT	-
24	D078-INSTALL-PLAN-REVISION	ELEMENT	-
25	D079-INSTALL-PLAN-PIECE-NUMBER	ELEMENT	-
26	D080-INSTALLATION-PLAN-QUANTIT	ELEMENT	-
27	D081-TYPE-MATL-REQMTS-DOC-CDE	ELEMENT	-
28	D082-MATERIAL-RQMTS-DOC-ITEM-N	ELEMENT	-
29	D083-EQUIP-COMP-MOD-IDENT-NR	ELEMENT	-
30	D084-TECH-MANUAL-NUMBER	ELEMENT	-
31	D085-REQUEST-NUMBER	ELEMENT	-
32	D086-MATERIAL-REQUIRE-DOCUMENT	ELEMENT	-
33	E001-APL-AEL-NUMBR	ELEMENT	-
34	E010-SRVC-APLICN-DSCRN	ELEMENT	-
35	E010A-SERVICE-APLICN-CODE	ELEMENT	-
36	E033-ACTION-CODE	ELEMENT	-
37	E052-PEI-LOCN-CODE	ELEMENT	-
38	E093-VLV-MK-ELEC-SYM-NR	ELEMENT	-
39	E126-EQUIPMENT-NAVY-SUPPORT-CA	ELEMENT	-
40	E127-WORK-CTR-RSPBL-FOR-COMPAR	ELEMENT	-
41	E128-WORK-CTR-RSPBL-FOR-EQUIPM	ELEMENT	-
42	E129-WORK-BREAKDOWN-STRUCTURE-	ELEMENT	-
43	E130-MAINTENANCE-INDEX-PAGE	ELEMENT	-
44	E132-ALLOWANCE-INDICATOR	ELEMENT	-
45	E133-ACCESS-NUMBER	ELEMENT	-
46	E134-TRANSACTION-ORIGNR-CODE	ELEMENT	-
47	E135-SPECIAL-REPORT-INDCR	ELEMENT	-
48	E136-CPSS-ITEM-INDICATOR	ELEMENT	-
49	E137-APPROV-AUTHRY-CODE	ELEMENT	-
50	E138-CHANGE-REQUEST-CATEGORY-C	ELEMENT	-
51	E139	ELEMENT	-
52	E140	ELEMENT	-
53	E141-PRCMT-SOURCE-DOC-ITEM-DUE	ELEMENT	-

Figure 7 - Example of a Consistency Check

PSA Version A5.1RS Jun 22, 1981 7:47:49
RS-A812

Data Activity Interaction Matrix Analysis

Data

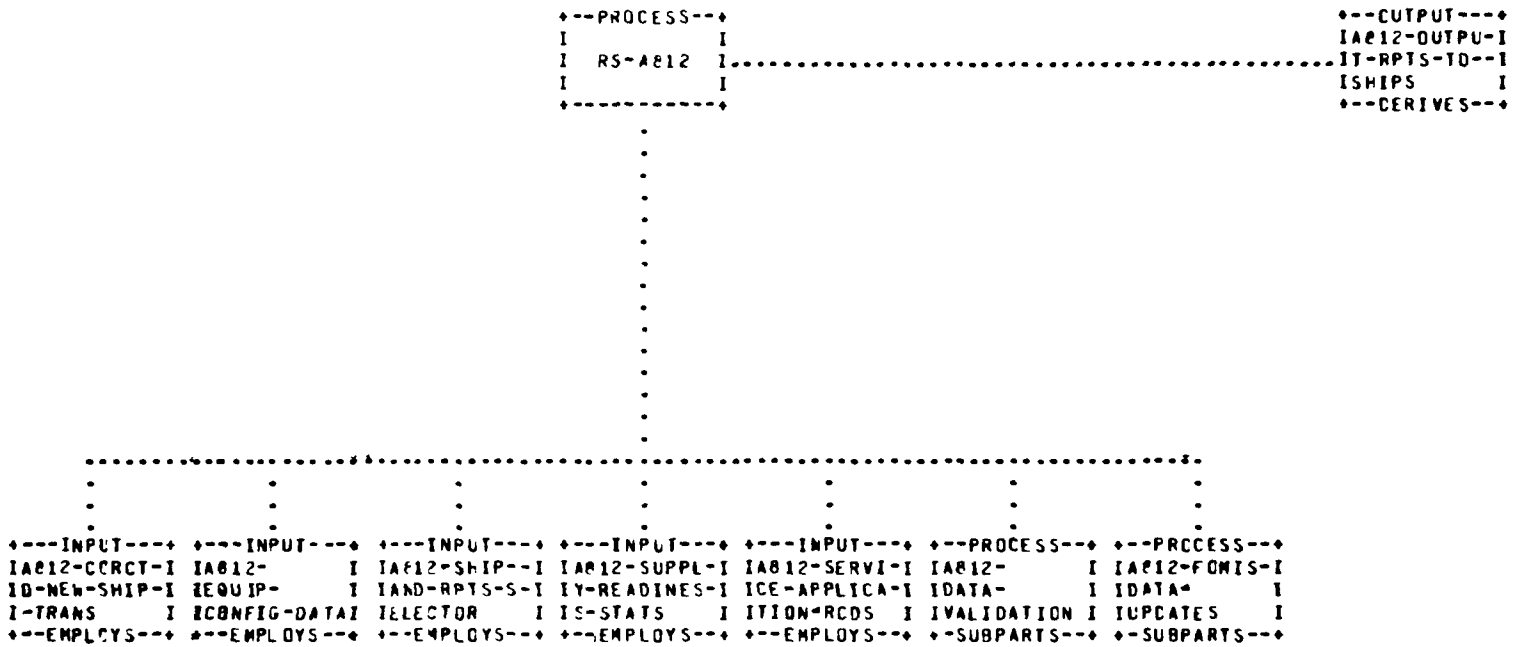
A812-CORCTD-NEW-SHIP-TRANS (INPUT) (Row 3)
not Received by any Activity
A812-OUTPUT-RPTS-TO-SHIPS (OUTPUT) (Row 6)
not Generated by any Activity
A812-SHIP-AND-RPTS-SELECTOR (INPUT) (Row 7)
not Received by any Activity
A812-SUPPLY-READINESS-STATS (INPUT) (Row 8)
not Received by any Activity
A812-SERVICE-APPLICATION-RCDS (INPUT) (Row 9)
not Received by any Activity
A812-EQUIP-CONFIG-DATA (INPUT) (Row 11)
not Received by any Activity

Figure 8 - Example of a Completeness Check

RS-A812

Picture Report

NAME=RS-A812



Continued on next page...

Figure 9 - Example of an RS Overview

```
PSA Version A5.1R5                              Jun 22, 1981   7:47:49
                                    RS-AR12

                          Formatted Problem Statement

    1 DEFINE GROUP                                  A812-UP-SPCC-EQUP-CONFG-REC;
    2     CONSISTS OF:
    3           C0C8B-MEC-CMPNT-EQPMT-FBM,
    4           C008D-MIL-ESENTLTY-CODE-CMPNT-,
    5           D0C9-APPLICATION-CODE,
    6           D029-APLICN-IDFCN-NR-ACTVY-COD,
    7           D083-EQUIP-COMP-MOD-IDENT-NR,
    8           D037-DATA-ORIGR-VALDN-CODE,
    9           E2C4-EQPMT-CMPNT-QTY-INSTLD-AT,
   10           E187-ACCOMPLSMT-INDICATOR,
   11           E207-STATISTICAL-VERIF-CODE,
   12           E2C8,
   13           E127-WORK-CTR-RSPBL-FOR-COMPAR;
   14     DERIVED BY:     A812-SPCC-RCD-UPDATES;
   15     EMPLOYED BY:    A812-REPORT-GENERATION;
   16
```

Figure 10 - Example of an RS Data Report

```
PSA Version A5.1R5                              Jun 22, 1981  10:16:52
                                    RS-AR12

                              Structure Report

    1    1 RS-A812                                          PROCESS
    2      2   A812-DATA-VALIDATION                         PROCESS
    3      2   A812-FOMIS-DATA-UPDATES                      PROCESS
    4      2   A812-SPCC-RCD-UPDATES                        PROCESS
    5      2   A812-REPORT-GENERATION                       PROCESS
```

Figure 11 - Example of RS Process Structure

```
PSA Version A5.1R5                    Jun 22, 1981   7:47:49
                                RS-A812

                        Formatted Problem Statement

    1 DEFINE PROCESS                              A812-DATA-VALIDATION;
    2     PART OF:        RS-A812;
    3     DERIVES:        A812-VALID-TRANSACTION;
    4     DERIVES:        A812-ERROR;
    5     EMPLOYS:        A812-CORCTD-NEW-SHIP-TRANS,
    6                     A812-FOMIS-DATA-BASE;
    7
    8 DEFINE PROCESS                              A812-FOMIS-DATA-UPDATES;
    9     PART OF:        RS-A812;
   10     DERIVES:        A812-UP-FOMIS-DATA-BASE;
   11     UPDATES:        A812-FOMIS-DATA-BASE;
   12     EMPLOYS:        A812-FOMIS-DATA-BASE,
   13                     A812-VALID-TRANSACTION;
   14
   15 DEFINE PROCESS                              A812-REPORT-GENERATION;
   16     PART OF:        RS-A812;
   17     DERIVES:        A812-OUTPUT-RPTS-TO-SHIPS;
   18     EMPLOYS:        A812-SHIP-AND-RPTS-SELECTOR,
   19                     A812-SUPPLY-READINESS-STATS,
   20                     A812-SERVICE-APPLICATION-RCDS,
   21                     A812-UP-FOMIS-DATA-BASE,
   22                     A812-UP-SPCC-EQUP-CONFG-REC,
   23                     A812-ERROR;
   24
   25 DEFINE PROCESS                              A812-SPCC-RCD-UPDATES;
   26     PART OF:        RS-A812;
   27     DERIVES:        A812-UP-SPCC-EQUP-CONFG-REC;
   28     UPDATES:        A812-SPCC-EQJP-CONFG-REC;
   29     EMPLOYS:        A812-EQUIP-CONFIG-DATA,
   30                     A812-SPCC-EQJP-CONFG-REC,
   31                     A812-FOMIS-DATA-BASE;
   32
   33 DEFINE PROCESS                              RS-A812;
   34     SYNONYMS ARE:   FOMIS;
   35     SUBPARTS ARE:   A812-DATA-VALIDATION,
   36                     A812-FOMIS-DATA-UPDATES,
   37                     A812-SPCC-RCD-UPDATES,
   38                     A812-REPORT-GENERATION;
   39     DERIVES:        A812-OUTPUT-RPTS-TO-SHIPS;
   40     EMPLOYS:        A812-CORCTD-NEW-SHIP-TRANS,
   41                     A812-EQUIP-CONFIG-DATA,
   42                     A812-SHIP-AND-RPTS-SELECTOR,
   43                     A812-SUPPLY-READINESS-STATS,
   44                     A812-SERVICE-APPLICATION-RCDS;
   45
```

Figure 12 - Example of RS Process Detail

PHASE 2. DESIGN GLOBAL LOGICAL DATA BASE

The output of this phase is to be a data base design which is global (i.e., it supports the data base requirements of all the subsystems) and logical (i.e., it is independent of any particular hardware or software environment). Ideally, this phase should be accomplished by developing a logical data base design for each subsystem, taking into consideration the functional requirements for the subsystem and the conceptual data structure, and then combining the subsystem designs to form a logical data base for the system. This procedure did not seem feasible in the ICP System redesign, because there was no conceptual data base to provide common names and no high-level structure to assure a reasonable degree of compatibility among the subsystem designs. The preferred approach and then the actual approach are discussed next.

DESIGN LOGICAL DATA STRUCTURE FOR SUBSYSTEMS

The output of this step should be a data structure which is independent of hardware and software and which supports a given subsystem. The data structure appears to the subsystem to represent a collection of application-oriented files - data of interest only to other subsystems is suppressed wherever possible. Note, however, that such data cannot always be suppressed: for example, a subsystem which prepares an invoice may have to add a control field which is of interest only to another subsystem. In general, it is possible to design top-down from the system to the subsystems, but not bottom-up by combining the logical data structures for the subsystems. The conceptual data structure is an indispensable input to this step to ensure the compatibility of the subsystems. The other input, the functional requirements for the subsystem, determines most, but not all, of the logical data structure for the subsystem. This step will probably be done in parallel with the development of the functional requirements for the subsystem and necessitate minor changes to the conceptual data structure.

DESIGN LOGICAL DATA STRUCTURE FOR THE SYSTEM

The output of this step should be a data structure which is independent of hardware and software, is more detailed than the conceptual data structure, but is limited in scope to data which is shared by different computer subsystems. The data may

be in a file (where they are controlled by one subsystem and accessed by a few subsystems), or in a data base (where they are controlled by a data base management system and are accessible by many subsystems). The logical data structure must, of course, provide all the data required by the computer subsystems. Feedback to both preceding phases is quite likely. The inputs to this phase are the conceptual data structure and the functional requirements for the system. The logical data structure may differ from the conceptual data structure in its period of applicability: the conceptual data structure should be valid as long as the organization remains in the same business, whereas the logical data structure will frequently change as different applications are performed by computer. The logical data structure for the system must be able, at any time, to satisfy all computerized data requirements at that time.

ACTUAL DESIGN OF THE GLOBAL LOGICAL DATA BASE

The actual steps taken were rather different from the ideal. First, a global logical data base (GLDB) structure was developed using a methodology that was both top-down (from real-world objects, concepts, and relationships) and bottom-up (from the DENs and non-standard data elements). The objective was to produce both the conceptual data structure and the logical data structure. The initial version of the GLDB proved to be extremely complex and much more expensive than expected. These problems seem to have been caused by the large number of analysts involved (as many as forty at the same time, on different RS's), by the lack of a conceptual data structure to serve as a guide, by the introduction of a large amount of detail (the data elements) at an early stage, and by the difficulty of resolving differences among the analysts. An extensive revision of the GLDB has been completed, with a great deal of improvement in both the simplicity and quality of the result. Part of the improvement is undoubtedly due to the availability of the first GLDB to serve as a guide, but a large part is due to the use of a much smaller group of people, working strictly top-down. This improved methodology consists of three steps:

1) Determine Entities and Relationships,
2) Define Boxes and Lines, and
3) Review and Revise GLDB Structure.

The first two steps are performed for each RS, and the third is performed as needed on the GLDB, which is a composite of all the RS's.

DETERMINE ENTITIES AND RELATIONSHIPS

This step determines the entities (real-world objects and concepts) and relationships with which a particular RS is concerned. The primary sources of guidance in determining the entities and relationships are RS Section 2, the System Summary; Section 3.2, the System Functions; Section 6.D, the Functional Flow Charts (see Figure 2 - Example of RS Structure); the RS Overview (Figure 9); the RS Data Report (Figure 10); the RS Process Structure (Figure 11); and the RS Process Detail (Figure 12). An entity may be a real-world object, such as a "supplier", or a real-world concept, such as an "invoice", and must have a meaningful name and an identifier (primary key) by which instances can be distinguished from one another. Reports are generally not entities, but the subjects of the reports are entities. Figure 13 shows the entities determined for one RS. Entities with the same identifiers are then combined into a single entity. Entity names are then changed to correspond to the names in the conceptual data structure (Figure 1). Relationships may be suggested by the processing described in the RS, but they are based on inherent properties of the data. Figure 14 is a greatly simplified diagram of the entities and relationships relevant to a particular RS. Note the combination of "issue", "receipt", etc. into "supply action." "Issue", "receipt", etc. may later become data subclasses - i.e., specific types of "supply action" - if they contain unique data elements.

activity
supply item
issue
receipt
adjustment
freeze/unfreeze
establish/select
restow
physical inventory request
(This list is for the exclusive use of the analyst working on this RS, and would normally be handwritten.)

Figure 13 - Example of an Initial List of Entities

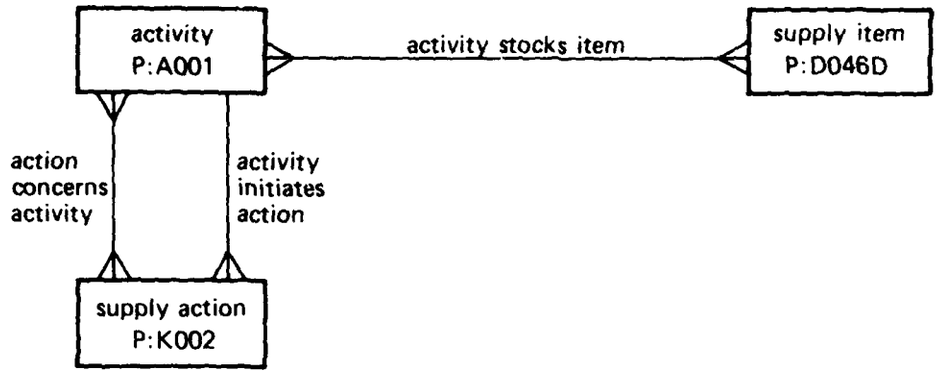


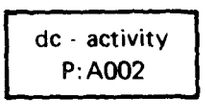
Figure 14 - Example of a Diagram of Entities and Relationships

(This diagram is for the exclusive use of the analyst working on this RS, and would normally be handwritten. The "P" indicates the primary key, or identifier.)

DEFINE BOXES AND LINES

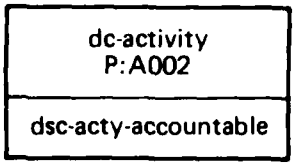
The next step is to define data clusters (represented on a data structure diagram by boxes) and relationships (represented by lines). There are five types of boxes, distinguished by whether the identifier is determined entirely from data within the data cluster (an entity that can exist independently of other entities), or determined in whole or in part by data in another box (in which case its existence depends on the existence of the other box):

1) A data class has an identifier (indicated by the "P") within the box:



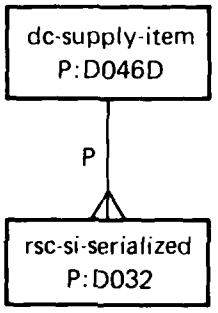
The box represents an independent entity.

2) A data subclass has a 1:1 relationship with a superior box, which provides the identifier:



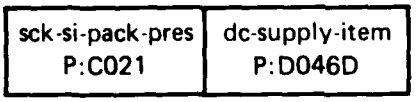
The inferior box represents a special case of the superior box.

3) A repeating subclass has an N:1 relationship with a superior box, which provides part of its identifier:



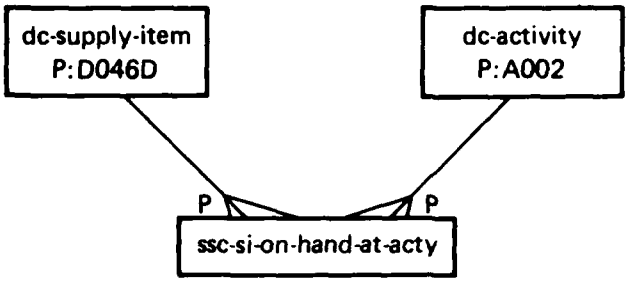
The box represents many instances and depends on the superior box.

4) A secondary key has a 1:N relationship with another box and identifies a subset of it:



The left-hand box represents a dependent entity (packaging and preservation rules) which is accessed only through the other entity.

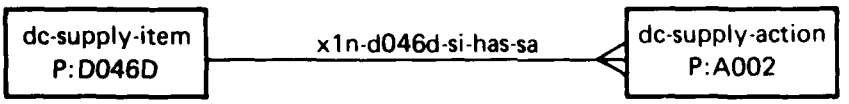
5) A shared subclass has two or more N:1 relationships with other boxes and is partially or completely identified by them:



The box represents an N:N relationship with possible intersection data (in this case, the quantity of an item at an activity). Its existence depends on the existence of the superior boxes.

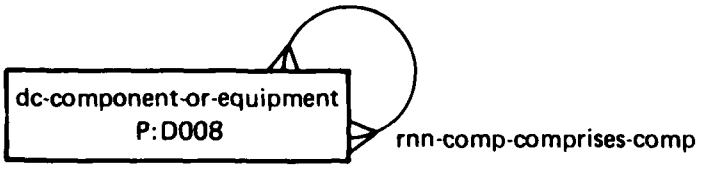
There are two types of lines, or relationships, other than those already introduced:

1) A cross-reference key represents a 1:1 or 1:N relationship between two boxes.



The connectivity and the symbolic key (a DEN) are indicated in the name of the relationship.

2) A recursive structure represents a 1:1, 1:N, or N:N relationship of a box to itself:



Note in all of these examples that naming conventions are very important. The effective use of PSL/PSA depends to a very great extent on the development and enforcement of strict naming conventions.

The graphic notation and PSL/PSA representation is based on two objectives: to capture all necessary information about the data, and to present that information in as simple a form as possible. The simplest form is a hierarchy of a data class, data subclasses, repeating subclasses, and secondary keys. Shared subclasses really belong to two or more hierarchies but are drawn under one hierarchy and represented in PSL as a subpart of one hierarchy. The result is a collection of interrelated hierarchies, each of which can be examined more or less independently. An example of one such structure is shown in Figure 15. A RELATION whose name begins with "rel" indicates a relationship with a shared subclass which has been placed in another hierarchy. For example, at line 12, "rel-si-stock-at-acty-acctbl-sp" indicates that there is a shared subclass called "ssc-si-stock-at-acty-acctbl-sp" in the "dc-supply-item" hierarchy. Figure 16 shows more detail about a particular entity, and Figure 17 shows more detail about a particular relationship.

This step has two results: a data structure diagram for a particular RS, and possible additions or modifications to the structure of the GLDB. A manually prepared, greatly simplified GLDB diagram is shown in Figure 18. Clearly manually prepared diagrams are almost impossible to maintain. The next step is intended to maintain the integrity of the GLDB structure as it expands in scope and detail.

REVIEW AND REVISE GLDB STRUCTURE

This review should be conducted frequently enough to detect and resolve problems before they have many side effects, yet not so frequently that only trivial problems are detected. Each data class is reviewed according to the following procedures:

1) Review the data class itself. Ensure that it has a meaningful name and description. Combine it with any previously reviewed data class that represents the same entity. Subdivide it if it represents two or more fundamentally different kinds of entities.

2) Review the hierarchy of data subclass, repeating subclasses, secondary keys, and recursive structures. Ensure that the identifier of each box is appropriate to its type.

PSA Version A5.1R58 Apr 19, 1992 13:47:38

Data-Classes

Structure Report

1	dc-activity	ENTITY
2	dsc-acty-accountable	ENTITY
3	dsc-acty-acctbl-icp	ENTITY
4	rsc-acty-acctbl-icp-item-mgr	ENTITY
5	rsc-acty-acctbl-icp-irc	ENTITY
6	dsc-acty-acctbl-stock-point	ENTITY
7	dsc-acty-acctbl-sp-sr	ENTITY
8	dsc-acty-acctbl-sp-sr-cyclic	ENTITY
9	dsc-acty-acctbl-sp-sr-rptg	ENTITY
10	dsc-acty-ac-sp-sr-ro-ammo-stor	ENTITY
11	rsc-acty-ac-sp-sr-ro-am-st-mag	ENTITY
12	rel-si-stock-at-acty-acctbl-sp	RELATION
13	dsc-acty-acft	ENTITY
14	dsc-acty-acft-flight-summary	ENTITY
15	rsc-acty-acft-flight-info	ENTITY
16	dsc-acty-acft-maint-summary	ENTITY
17	rsc-acty-acft-maint-info	ENTITY
18	sck-acty-acft-type-mod-series	ENTITY
19	xln-a002-acty-has-acft-type	RELATION
20	ssc-acty-acft-uses-si-serial	ENTITY
21	dsc-acty-address	ENTITY
22	dsc-acty-avcal-ves-or-shore	ENTITY
23	dsc-acty-avcal-ctl	ENTITY
24	dsc-acty-avcal-sum	ENTITY
25	dsc-acty-config-stts-acctg	ENTITY
26	rsc-acty-hardware-sys-command	ENTITY
27	rsc-acty-wkctr	ENTITY
28	rsc-acty-wkctr-mnt-tsk	ENTITY
29	rel-mnt-tsk-cotly-at-acty-wctr	RELATION
30	sck-acty-major-command	ENTITY
31	ssc-acty-has-skill-available	ENTITY
32	ssc-acty-ima-has-si-bcm	ENTITY
33	ssc-acty-rcvs-tech-info-issued	ENTITY
34	ssc-acty-uses-approp-issued-to	ENTITY
35	dsc-acty-cosal-ves-or-cb	ENTITY
36	dsc-acty-cosal-const-battalion	ENTITY
37	dsc-acty-cosal-cb-ctl	ENTITY
38	dsc-acty-cosal-ves-control	ENTITY
39	dsc-acty-cosal-ves-reqn-ctl	ENTITY
40	dsc-acty-cosal-ves-summary	ENTITY
41	dsc-acty-cosmal-grl-country	ENTITY
42	dsc-acty-cosmal-grl-ctry-ctl	ENTITY
43	dsc-acty-cosmal-grl-eq-list	ENTITY
44	dsc-acty-cosmal-grl-eq-lst-sum	ENTITY
45	dsc-acty-isea	ENTITY
46	dsc-acty-overhaul-point	ENTITY
47	dsc-acty-trident	ENTITY
48	dsc-acty-ves	ENTITY
49	dsc-acty-ves-fomis	ENTITY
50	dsc-acty-ves-fomis-document	ENTITY
51	rsc-acty-ves-vamosc	ENTITY
52	dsc-acty-ves-vamosc-cains	ENTITY
53	dsc-acty-ves-vamosc-cmc	ENTITY

Figure 15 - Example of GLDB Structure

```
PSA Version A5.1R58                          Apr 19, 1982  13:47:39
                                Data-Classes

                          Formatted Problem Statement

 1 DEFINE ENTITY                                    dc-activity;
 2      SYNONYMS ARE:  DC-ACTIVITY-SHORE-STATION;
 3      DESCRIPTION;
 4 Organizational units performing mission related functions such as shore
 5 stations, ships, countries, aircraft squadrons, aircraft, and
 6 Tycoms.;
 7      SUBPARTS ARE:  dsc-acty-accountable,
 8                     dsc-acty-acft,
 9                     dsc-acty-address,
10                     dsc-acty-avcal-ves-or-shore,
11                     dsc-acty-config-stts-acctg,
12                     rsc-acty-hardware-sys-command,
13                     rsc-acty-wkctr,
14                     sck-acty-major-command,
15                     ssc-acty-has-skill-available,
16                     ssc-acty-imaa-has-si-bcn,
17                     ssc-acty-rcvs-tech-info-issued,
18                     ssc-acty-uses-approp-issued-to,
19                     dsc-acty-cosal-ves-or-cb,
20                     dsc-acty-cosmal-grl-country,
21                     dsc-acty-isea,
22                     dsc-acty-overhaul-point,
23                     dsc-acty-trident,
24                     dsc-acty-ves,
25                     dsc-acty-3m-maint-stat,
26                     dsc-acty-ovhl-yard-ship-yard;
27      IDENTIFIED BY: A002,
28                     A001B,
29                     A001D,
30                     D035B;
31      LEFT PART OF:  rln-acty-performs-paa-function,
32                     rel-comp-mnt-tsk-cpbly-at-acty,
33                     rel-comp-used-at-acty,
34                     rel-si-due-in-at-acty,
35                     rel-si-pub-itm-on-hand-at-acty,
36                     rel-si-allowed-to-acty,
37                     rel-si-on-hand-at-acty,
38                     rel-si-on-hand-at-acty-avcal,
39                     rel-si-on-hand-at-acty-cosal,
40                     rel-si-inventory-data-at-acty,
41                     rel-si-qty-iss-or-recd-by-acty,
42                     rel-si-used-at-acty,
43                     rel-si-rep-on-hand-at-acty,
44                     rln-acty-home-port-to-acty-ves,
45                     rln-acty-plan-yard-to-acty-ves,
46                     rln-acty-has-acty-acft;
47      RIGHT PART OF: rln-acty-performs-paa-function,
48                     rnn-acty-ves-built-at-acty,
49                     xln-a002-acty-ctls-tech-info,
50                     xln-a001b-acty-submits-sa,
51                     xln-a001d-acty-receives-sa,
52                     xln-t002-acty-ac-icp-im-ctl-si,
53                     xln-a002-acty-dsgtd-by-clin,
54                     xln-a002-acty-isea-ctls-si,
55                     xln-a002-acty-from-subj-of-sa,
```

Figure 16 - Example of Entity Detail

```
PSA Version A5.1R5B                    Apr 19, 1982  13:47:38
                                  Data-Classes

                          Formatted Problem Statement

    1 DEFINE RELATION                                  x1n-1046d-si-has-sa;
    2      SYNONYMS ARE:  SSC-ITEM--ACTION;
    3      DESCRIPTION;
    4 Each individual item of supply and how it is affected by the various
    5 types of supply actions.;
    6      LEFT IS:       dc-supply-action;
    7      RIGHT IS:      dc-supply-item;
    8
```

Figure 17 - Example of Relationship Detail

3) Review the shared subclasses and cross-reference keys in the hierarchy. Ensure that the identifiers of shared subclasses are appropriate. Ensure that the relationships are meaningful.

4) Resolve any problems that could not be resolved within an individual hierarchy.

5) Eliminate redundant relationships.

Final documentation may be developed when all RS's have been completed. The use of PSL/PSA to represent the evolving structure should eliminate the need for any manually prepared documentation.

CRITIQUE

The problems encountered in this phase could have been avoided by:

1) Using a smaller, better trained group of analysts.

2) Having guidance available in the form of a good conceptual data structure.

3) Adhering to a strict top-down design methodology, with data elements assigned only after the high-level GLDB structure had been completed for all RS's.

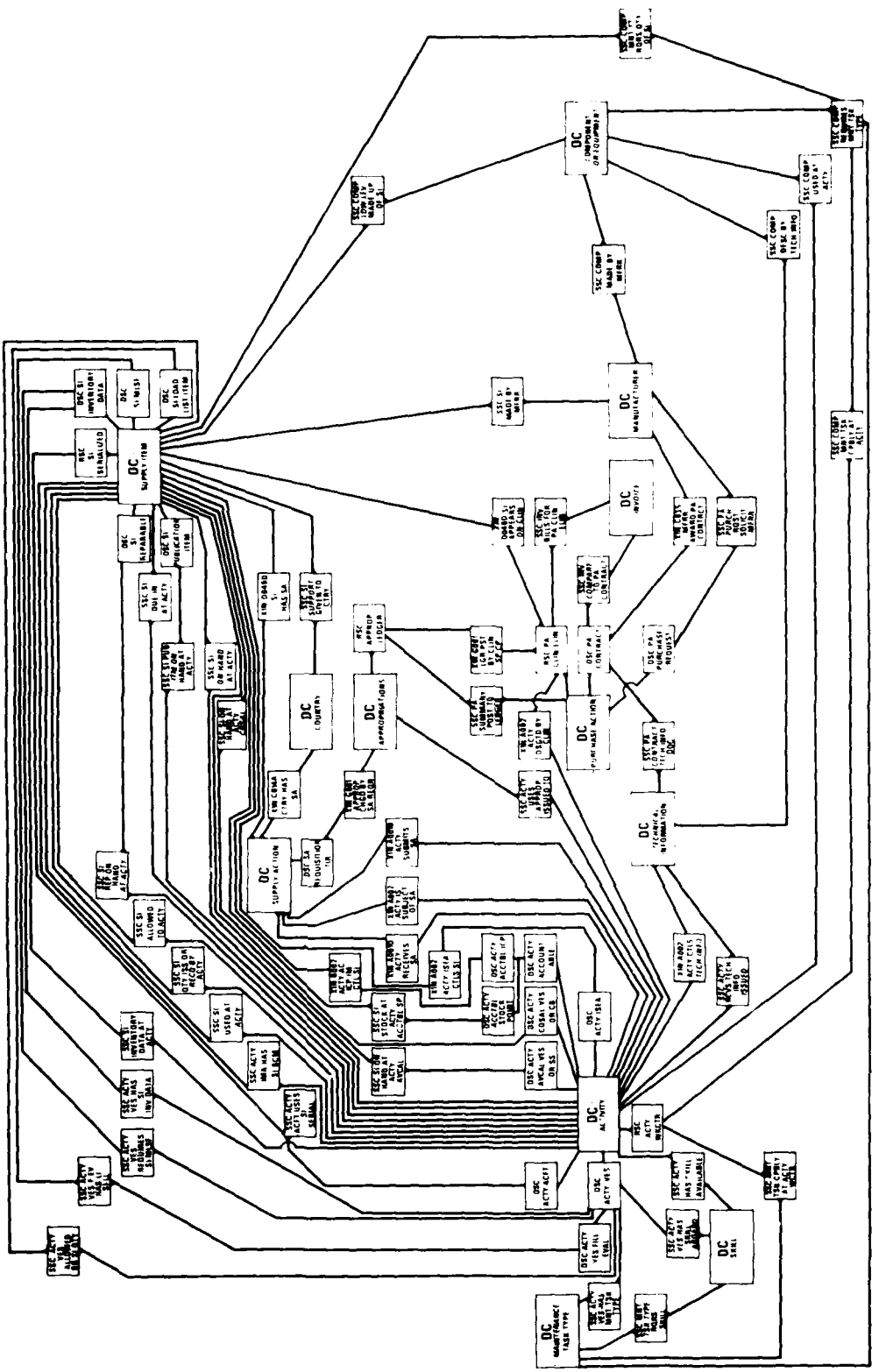


Figure 18 - Example of a GLDB Diagram

PHASE 3. DEFINE DATA BASE PROCESSES

The output of this phase is to be both a validation of the GLDB structure and the development of the workload on it. The following seven steps are performed for each RS:

1) Map the RS Data into the GLDB Structure
2) Split or Combine RS Processes
3) Assign Data to File or Data Base
4) Review RS Processes
5) Diagram Data Base Processes
6) Represent Data Base Processes in PSL/PSA
7) Review and Revise GLDB Structure.

A small, well-trained group of NAVSUP analysts performed the first five steps for six sample RS's. Although the steps are tedious, the results seem to be quite satisfactory. The sixth step was performed on three of the six RS's. As expected, the GLDB structure did require revision.

MAP THE RS DATA INTC THE GLDB STRUCTURE

This step requires the analysis of each INPUT, OUTPUT, or GROUP in the RS to determine the entities and relationships in the GLDB required to represent it. The analysis is initially done top-down - i.e., the INPUT, OUTPUT, or GROUP is represented first in terms of data class hierarchies, and then in the specific entities and relationships within the hierarchy. The result is verified by checking to ensure that each data element in the INPUT, OUTPUT, or GROUP is represented somewhere in the entities and relationships.

SPLIT OR COMBINE RS PROCESSES

First, this step requires the analysis of each process within the RS to determine what entities and relationships are used, derived, or updated by it. Second, each process is split into simpler processes if it involves a delay in data base interaction (e.g., if it waits for verification by a clerk), or if it uses, derives, or updates different sets of entities and relationships at different times (e.g., if it uses different data at the end of the week and the end of the month). The process

is split because it represents two (or more) different processes as far as the data interactions are concerned. Third, successive processes are combined if they occur at the same frequency, if there are no delays, and if they interact with the same entities and relationships. The processes are combined in order to more accurately represent the workload - instead of two sequences of data base interactions, they really represent a single sequence.

ASSIGN DATA TO FILE OR DATA BASE

This step involves determining whether an entity or relationship should be stored in a private file or in a data base. The data should be stored in a private file if any of the following are true:

1) The data are of interest to only a single process and therefore need not be shared in a data base.

2) The data are transitory and would not exist long enough to be relevant to other processes.

3) The data are incomplete, as in a partially completed update, and therefore could not be used by other processes.

4) The data consist entirely of references, or keys, to other data, are of interest to only one process, and are therefore irrelevant to other processes.

The data should be stored in the data base if all of the following are true:

1) The data are of interest to many processes and should therefore be shared.

2) The data are sufficiently long-lived to have many uses.

3) The data are complete.

4) The data are descriptive of the real world.

REVIEW RS PROCESSES

The next step is to eliminate from consideration all data that have been assigned to private files and all processes that do not interact with the data base. The remaining processes should then be reviewed for possible additional combination:

1) Successive processes are combined as in the second step if they really represent two parts of a sequence of data base interactions.

2) Processes with identical data base interactions are combined, with suitable adjustment of frequency of occurrence, since they are indistinguishable as far as the workload is concerned.

DIAGRAM DATA BASE PROCESSES

This step involves the construction of a diagram showing the path through the data base traversed by each data base process. Figure 19 shows a part of such a diagram. The rectangles represent entities, the arrows represent relationships, the circled numbers indicate the order of processing, the DEN on an arrow indicates a retrieval key, the number near an arrow head represents the number of instances retrieved, and the number inside the rectangle represents the number of instances actually used (some of the instances may have been retrieved only to check for some value). A DEN inside a rectangle represents the order in which the retrieved instances are to be used, and an "A" indicates that an entity is needed only to access another entity and that it has no data needed by the process (not shown in the example). A "U" indicates that an entity or relationship is to be updated.

If it is impossible to construct a diagram because either a required relationship or a required entity is not in the GLDB structure, the GLDB structure obviously must be revised.

REPRESENT DATA BASE WORKLOAD IN PSL/PSA

Figure 20 shows an example of part of the structure of processes needed to represent the data base workload. The highest level subsumes the workload at the Ships Parts Control Center (SPCC) in Mechanicsburg, Pennsylvania; a similar structure exists for the Aviation Supply Office (ASO) in Philadelphia. The second level represents the different applications. The third level consists of the data base processes. The fourth level describes each step in the path through the data base.

Figure 21 shows a generalized example of a diagram of a data base process. Note that "entity-e1" (possibly a data class) has subparts, "entity-e2", "entity-e3", etc. (data subclasses, repeating subclasses, or secondary keys). Figure 22 shows the representation in PSL of the diagram. Lines 2-6 indicate the number of instances examined and the number accepted. Line 7 represents the "A" above the box. Lines 9-10 and 15-17 indicate volatility data not on the diagram. Lines 11 and 13 show the data actually used (there could be many data subclasses, repeating subclasses, or secondary keys), and lines 12 and 14 show the DENs used as the sort criteria (there could be many DENs). Line 18 identifies the line into the box, and line 11

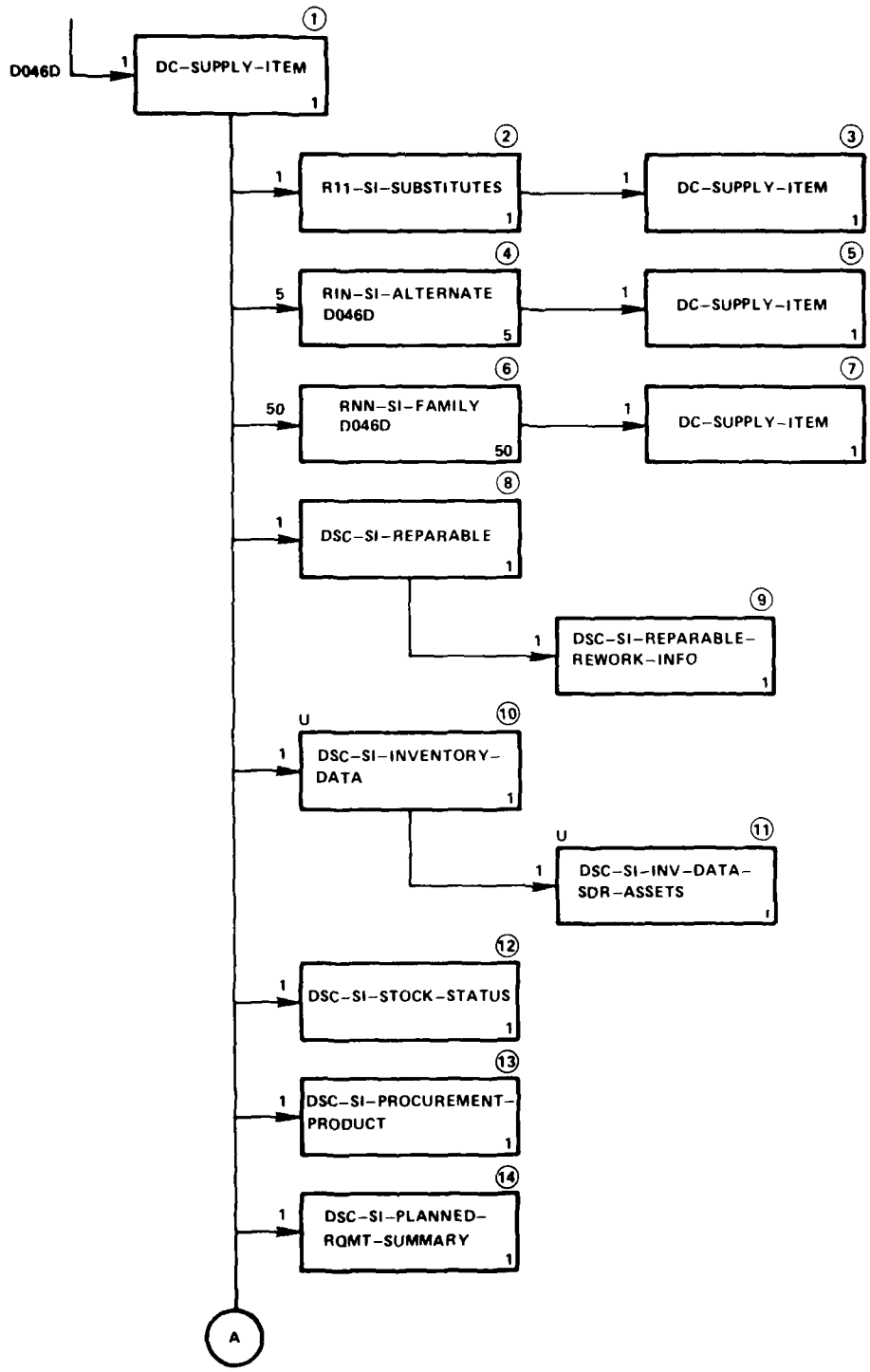


Figure 19 - Example of a Diagram of a Data Base Process

PSA Version A5.1R58 Apr 19, 1982 13:47:38

Data-Classes

Structure Report

```
 1    spcc-workload                                  PROCESS
 2       spcc-a213-dbp                               PROCESS
 3          spcc-a213-1                              PROCESS
 4             spcc-a213-1-1                         PROCESS
 5             spcc-a213-1-2                         PROCESS
 6             spcc-a213-1-3                         PROCESS
 7             spcc-a213-1-4                         PROCESS
 8             spcc-a213-1-5                         PROCESS
 9             spcc-a213-1-6                         PROCESS
10             spcc-a213-1-7                         PROCESS
11             spcc-a213-1-8                         PROCESS
12             spcc-a213-1-9                         PROCESS
13             spcc-a213-1-10                        PROCESS
14             spcc-a213-1-11                        PROCESS
15             spcc-a213-1-12                        PROCESS
16             spcc-a213-1-13                        PROCESS
17             spcc-a213-1-14                        PROCESS
18             spcc-a213-1-15                        PROCESS
19             spcc-a213-1-16                        PROCESS
20          spcc-a213-2                              PROCESS
21             spcc-a213-2-1                         PROCESS
22             spcc-a213-2-2                         PROCESS
23             spcc-a213-2-3                         PROCESS
24             spcc-a213-2-4                         PROCESS
25             spcc-a213-2-5                         PROCESS
26             spcc-a213-2-6                         PROCESS
27             spcc-a213-2-7                         PROCESS
28             spcc-a213-2-8                         PROCESS
29             spcc-a213-2-9                         PROCESS
30             spcc-a213-2-10                        PROCESS
31             spcc-a213-2-11                        PROCESS
32             spcc-a213-2-12                        PROCESS
33       spcc-a221-dbp                               PROCESS
34          spcc-a221-1                              PROCESS
35             spcc-a221-1-1                         PROCESS
36             spcc-a221-1-2                         PROCESS
37             spcc-a221-1-3                         PROCESS
38             spcc-a221-1-4                         PROCESS
39             spcc-a221-1-5                         PROCESS
40             spcc-a221-1-6                         PROCESS
41             spcc-a221-1-7                         PROCESS
42          spcc-a221-2                              PROCESS
43             spcc-a221-2-1                         PROCESS
44             spcc-a221-2-2                         PROCESS
45             spcc-a221-2-3                         PROCESS
46             spcc-a221-2-4                         PROCESS
47             spcc-a221-2-5                         PROCESS
48             spcc-a221-2-6                         PROCESS
49             spcc-a221-2-7                         PROCESS
50             spcc-a221-2-8                         PROCESS
51             spcc-a221-2-9                         PROCESS
52             spcc-a221-2-10                        PROCESS
53             spcc-a221-2-11                        PROCESS
54             spcc-a221-2-12                        PROCESS
55             spcc-a221-2-13                        PROCESS
```

Figure 20 - Example of Workload Structure

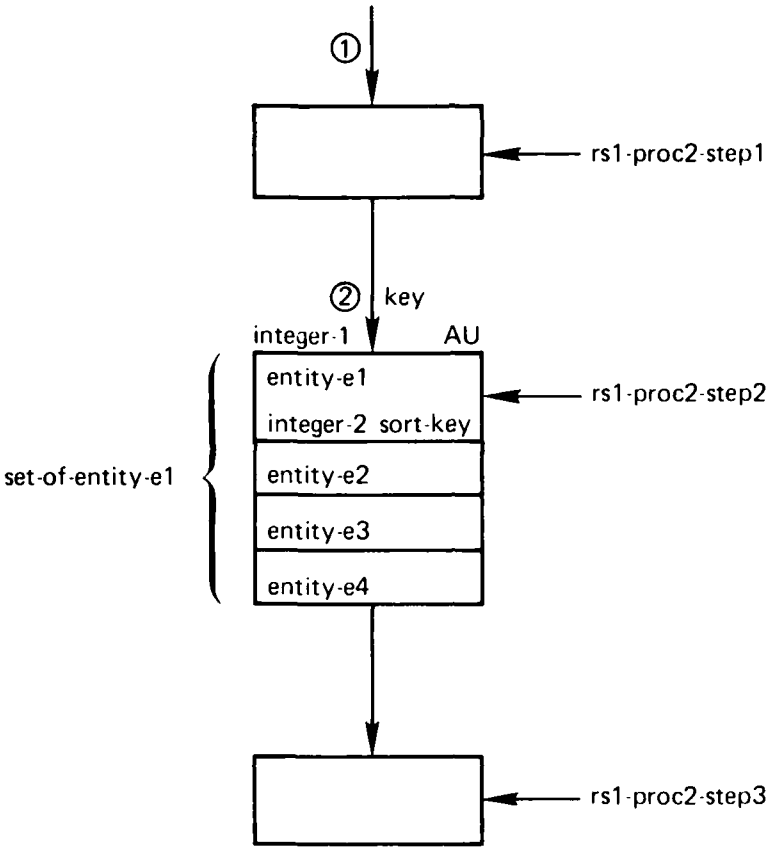


Figure 21 - Generalized Example of a Diagram of a Data Base Process

the basic entity. Lines 19-21 represent the frequency with which the step occurs, and lines 22-23 indicate its successor and predecessor.

Figure 23 shows the PSL/PSA representation of part of Figure 19. Boxes 1, 8, 9, 10, 11, 12, 13, 14 and others not shown in Figure 19 are DERIVED (accessed) simultaneously because they are parts of the same basic entity. Figure 24 shows the PSL/PSA representation of Box 2 of Figure 19. Note that "r11-si-substitutes" on Figure 19 has been expanded into a PSL RELATION ("rej-si-substitutes") to a dummy record ("dum-si-substitutes") and a RELATION ("rem-si-substitutes") from the dummy record. The inverse RELATIONs, which are not needed in the example, are "rek-si-substitutes" and "rel-si-substitutes." All recursive relationships and shared subclasses must be similarly expanded to explicitly represent all the implied relationships.

```
PSA Version A5.1R5B                          Apr 19, 1982  13:47:38
                          Sample-Workload

                     Formatted Problem Statement

 1 DEFINE PROCESS                                  rs1-proc2-step2;
 2      ATTRIBUTES ARE:
 3           number-first-retrieved
 4                          integer-1,
 5           number-finally-used
 6                          integer-2;
 7      KEYWORDS ARE:   "used-only-to-access-other-data";
 8      PART OF:        rs1-proc2;
 9      CREATES:
10           relation-r1;
11      DERIVES:        entity-e1
12        USING:        sort-key;
13      DERIVES:        entity-e2
14        USING:        sort-key;
15      DESTROYS:
16           relation-r2;
17      MODIFIES:       number-modified entity-e4 IN set-of-entity-e1;
18      EMPLOYS:        relation-to-get-to-e1;
19      HAPPENS :
20           some-frequency
21            TIMES-PER      year;
22      COMES BEFORE:   rs1-proc2-step3;
23      COMES AFTER:    rs1-proc2-step1;
24
```

Figure 22 - Generalized Example of Workload Detail

REVIEW AND REVISE GLDB STRUCTURE

The final step is to determine whether any entities are frequently accessed only to access other entities and relationships. If so, such accesses may be eliminated by the introduction of different relationships to provide direct access to the needed data. PSA provides a report (Element-Process Utilization) which can be used to determine how frequently each entity is accessed by processes with the KEYWORD "used-only-to-access-other-data."

```
PSA Version A5.1H58                          Apr 20, 1982   9:56:29
                              Data-Classes

                        Formatted Problem Statement

 1 DEFINE PROCESS                                    spcc-a221-2-4;
 2      ATTRIBUTES ARE:
 3           number-first-retrieved
 4                                1,
 5           number-finally-used
 6                                1;
 7      PART OF:          spcc-a221-2;
 8      DERIVES:          dc-supply-item;
 9      DERIVES:          dsc-si-reparable;
10      DERIVES:          rsc-si-reparable-rework-info;
11      DERIVES:          dsc-si-inventory-data;
12      DERIVES:          dsc-si-inv-data-sdr-assets;
13      DERIVES:          dsc-si-stock-status;
14      DERIVES:          dsc-si-procurement-product;
15      DERIVES:          dsc-si-planned-rqmt-summary;
16      DERIVES:          dsc-si-physical-characteristic;
17      DERIVES:          dsc-si-cataloging-mgt-data;
18      DERIVES:          dsc-si-transportation-data;
19      DERIVES:          dsc-si-disposable;
20      UPDATES:          dsc-si-inventory-data;
21      UPDATES:          dsc-si-inv-data-sdr-assets;
22      EMPLOYS:          D0460;
23      HAPPENS :
24           550600
25             TIMES-PER       month;
26      COMES BEFORE:     spcc-a221-2-2,
27                        spcc-a221-2-4,
28                        spcc-a221-2-6,
29                        spcc-a221-2-8,
30                        spcc-a221-2-10,
31                        spcc-a221-2-12,
32                        spcc-a221-2-13,
33                        spcc-a221-2-14,
34                        spcc-a221-2-16,
35                        spcc-a221-2-17,
36                        spcc-a221-2-18,
37                        spcc-a221-2-19,
38                        spcc-a221-2-20;
39
```

Figure 23 - Example of Workload Detail

```
PSA Version A5.1R5B                          Apr 20, 1982   9:56:29
                                  Data-Classes

                          Formatted Problem Statement

 1 DEFINE PROCESS                                        spcc-a221-2-2;
 2      ATTRIBUTES ARE:
 3           number-first-retrieved
 4                          1,
 5           number-finally-used
 6                          1;
 7      PART OF:          spcc-a221-2;
 8      DERIVES:          dum-si-substitutes;
 9      EMPLOYS:          rej-si-substitutes;
10      HAPPENS :
11           550600
12             TIMES-PER       month;
13      COMES BEFORE:     spcc-a221-2-3;
14      COMES AFTER:      spcc-a221-2-1;
15
16 DEFINE PROCESS                                        spcc-a221-2-3;
17      ATTRIBUTES ARE:
18           number-first-retrieved
19                          1,
20           number-finally-used
21                          1;
22      PART OF:          spcc-a221-2;
23      DERIVES:          dc-supply-item;
24      EMPLOYS:          rem-si-substitutes;
25      HAPPENS :
26           550600
27             TIMES-PER       month;
28      COMES AFTER:      spcc-a221-2-2;
29
```

Figure 24 - Continuation of Workload Detail

PHASE 4. DESIGN PHYSICAL DATA BASE

The output of this phase is the design for a physical data structure which will be hardware and software dependent and which will support all the subsystems at a particular time. The physical data structure may be initially limited to the needs of one or two subsystems but will grow as more subsystems are implemented. Since the functional requirements are based on the logical data structures, existing subsystems will be unaffected as new data are added to the physical data base. The input to this phase consists of the logical data structure for the system, the workload defined by the functional requirements for the subsystems, and hardware and software specifications. In addition to satisfying all computerized data requirements, the physical data structure must be efficient - note that this is the first phase in which efficiency is an issue.

There are six steps in this phase:

1) Simplify Data Base Structure and Processes
2) Determine Sizes, Volumes, and Volatilities
3) Form Canonical Records
4) Convert to Physical Level
5) Design Physical Structure
6) Analyze Results and Iterate.

The first two steps are merely preparation for the use of a computer-aided data base design system,[4,5,6,7] which does the computational work involved in the third through fifth steps.

SIMPLIFY DATA BASE STRUCTURE AND PROCESSES

The objective of this step is to reduce the complexity of the GLDB and the workload enough to apply the design system described in steps 3 through 5. The complexity of the ICP System design would be too great to be handled by the design system: hundreds of entities would consist of thousands of data elements, and would be retrieved by thousands of different processes.

The GLDB structure can be greatly simplified by treating all data subclasses and secondary keys as if they were data elements (i.e., descriptors which cannot be further subdivided). The only DENs which need appear in the GLDB structure are those

which are used as keys to retrieve or sort data. (Note that the data base processes were defined in these terms in the previous phase.)

The data base processes can be simplified by combining those that represent identical paths through the GLDB and eliminating those that occur relatively infrequently.

DETERMINE SIZES, VOLUMES, AND VOLATILITIES

This step is very simple: it involves only the determination of the size (in characters), the number of instances, and the volatility (number of creations, deletions, and modifications) of each entity. The size is merely the sum of the sizes of the data elements in the entity.

FORM CANONICAL RECORDS

This step is performed primarily by the data base design system, although there can be some interaction with the data base designer. The design system uses the GLDB structure to generate different collections of "canonical" records. A canonical record is a set of data elements and relationship pointers (symbolic or physical) which will be physically represented by a (possibly segmented) physical record. Each collection of canonical records represents a different way of combining entities and relationships to include the entire GLDB. For example, a repeating subclass subordinate to a data class could be represented by a repeating group within the data class or by a separate canonical record. If represented by a separate canonical record, the relationship between the data class and the repeating subclass could be represented by symbolic or physical pointers, or both, either from the data class to the repeating subclass, or from the repeating subclass to the data class, or both ways. The design system includes various heuristics for reducing the number of canonical records; otherwise, the number of different combinations of canonical records would be far too great to be manageable. The data base designer can override the heuristics and add other collections of canonical records, if desired.

CONVERT TO PHYSICAL LEVEL

This step involves the conversion of the original workload, sizes, volumes, and volatilities, which were defined for the GLDB, to the corresponding parameters for each collection of canonical records. To continue the previous example, if the

repeating subclass becomes a repeating group in the data class, then any transversal of the original relationship between the two will vanish at this step, and the size and volatility of the data class will change to reflect the additional size and volatility of the repeating subclass. The conversion is performed by the design system.

DESIGN PHYSICAL STRUCTURE

At this point the design system requires parameters describing the hardware and software environment: sizes, speeds, the relative costs of retrieval time, update time, storage space, availability of record segmentation, etc. For each collection of canonical records, the design system determines a set of physical record structures and access paths with near minimal cost and provides an evaluation of performance. The design system can also evaluate a physical structure proposed by the data base designer. Because heuristics are used to limit the number of possible logical and physical structures generated and evaluated, it is impossible to guarantee that the design system will produce an optimal structure; results to date, however, indicate that the structures are close to optimal for reasonable situations.

ANALYZE RESULTS AND ITERATE

The final step is the responsibility of the data base designer. The hardware, cost, and software parameters may be changed, and new physical structures generated and evaluated. For example, the data base designer may be interested in comparing the performance of different data base management systems.

PHASE 5. SIMULATE DATA BASE OPERATION

The data base design system is intended to generate and evaluate a large number of reasonable physical data structures, given simplifying assumptions about the workload (e.g., that data base processes are distributed evenly throughout the working day), the hardware (e.g., that retrieval time, update time, and storage are the scarce resources), etc. These are reasonable assumptions in many cases, but they must be verified by more detailed analysis. This detail will be provided by a data base simulator[8] based on the Extendable Computer System Simulator (ECSS),[9] a preprocessor for SIMSCRIPT II.5.[10] The simulator is currently being developed and verified for ISDNLS by the Federal Computer Performance Evaluation and Simulation Center (FEDSIM). ECSS provides special, pre-coded models for hardware and operating system simulation; the data base simulator will add models for the special case of a data base and data base management system (currently limited to CODASYL). The result will provide a capability for tracing simulated retrievals and updates to determine peak load, channel or memory contention, operating system performance, etc. This level of detail would clearly not be possible when large numbers of data structures were being evaluated, but it is reasonable for evaluation of a small number which seem to be close to optimal. Also, the detailed simulation allows for determination of the sensitivity of the data base performance to small changes in structure, workload, hardware, and software. The quick, approximate data base design system and the slow, detailed simulator appear to complement each other very well.

PHASE 6. DESIGN OPERATIONAL SUBSYSTEMS

The methodology currently does not cover this phase, but some general remarks are nevertheless appropriate. This final output of the design process should be a specification of the way in which each subsystem will be implemented. If operational specifications are produced without going through the preceding phases, the choices which can be made by designers will be reduced, and system costs will increase and system effectiveness decrease. Furthermore, the large amount of detail involved in all the operational specifications may overwhelm the designers, so that many mistakes will be made, particularly in the interfaces. Finally, flexibility will certainly be reduced without the benefit of the preceding phases.

Much of the difficulty encountered in the ICP System redesign can be traced to writers who wrote operational specifications, rather than functional requirements, in the RS's; more guidance and training would have led to a better result in less time at lower cost.

CONCLUSIONS

A great deal of unnecessary effort could have been avoided in the ICP System redesign by adhering to the following guidelines:

1) Follow a strict top-down design sequence, as outlined in Phases 0 through 6

2) Perform the design phases with a small number of highly trained people.

The use of PSL/PSA has proven successful. The data base design system and data base simulator are expected to be equally successful.

REFERENCES

1. Teichroew, D. and E.A. Hershey, III, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 41-48 (1977).

2. Reference manuals on PSL/PSA, ISDOS Project, Department of Industrial and Operations Engineering, Dr. D. Teichroew, The University of Michigan, Ann Arbor, Michigan 48109, telephone (313) 763-2238.

3. Ross, D.T. and K.E. Schoman, Jr., "Structured Analysis for Requirements Definition," IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 6-15 (1977).

4. Carlis, J.V., "An Investigation into the Modeling and Design of Large, Logically Complex, Multi-User Databases," PhD Thesis, University of Minnesota (December 1980).

5. March, S.T. and J.V. Carlis, "A Computer Aided Database Design Methodology," Management Information Systems Research Center Report Number MISRC-TR-81-01, University of Minnesota (September 1980).

6. Carlis, J.V. and S.T. March, "A Computerized Database Design System Version 5: User's Manual," Management Information Systems Research Center Report Number MISRC-TR-81-02, University of Minnesota (November 1980).

7. Carlis, J.V. and S.T. March, "A Computerized Database Design System Version 5: System Manual," Management Information Systems Research Center Report Number MISRC-TR-81-03, University of Minnesota (November 1980).

8. Aitken, J.A. and H.T. Hsu, "On the Simulation of Network Model Data Base Management Systems," NBS Special Publication 500-65, National Bureau of Standards Computer Performance Evaluation User's Group Proceedings (October 1980).

9. Kosy, D.W., "The ECSS II Computer Language for Simulating Computer Systems," Report R-1895-GSA, the RAND Corporation (December 1975).

10. Kiviat, P.J. et al., "SIMSCRIPT II.5 Programming Language," C.A.C.I., Inc., Los Angeles (1973).

INITIAL DISTRIBUTION

Copies		
8	NAVSUP	
	1	04R
	1	04R1
	1	033
	5	033A
8	NFMSO	
	1	9R
	1	9RA
	5	9R2
	1	944
12	DTIC	

CENTER DISTRIBUTION

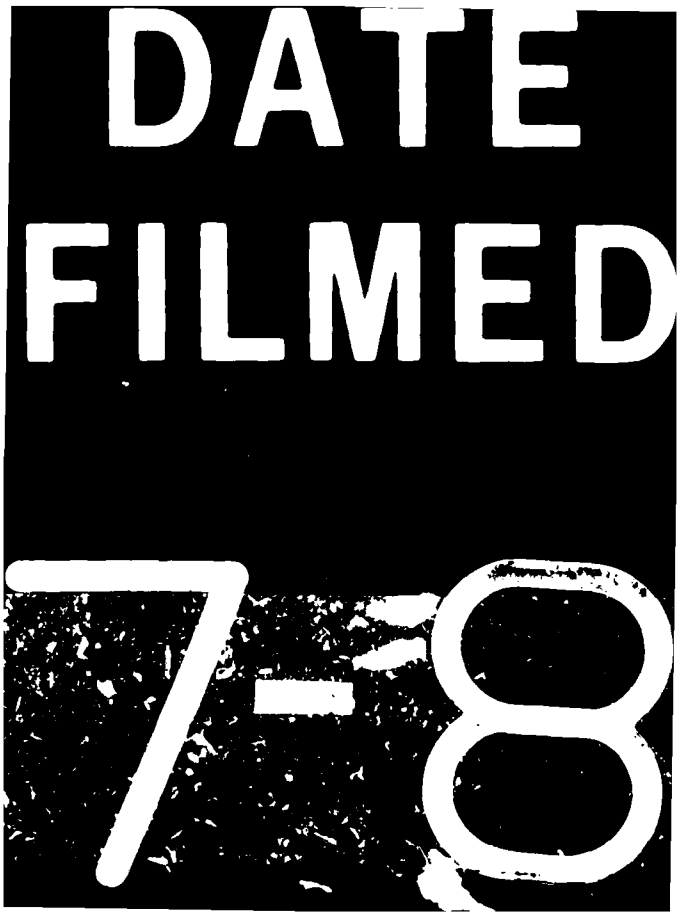
Copies	Code	Name
1	18	G. Gleissner
2	1809.3	D. Harris
1	182	A. Camara
30	1821	D. Jefferson
1	1822	T. Rhodes
1	1824	S. Berkowitz
1	1826	L. Culpepper
10	5211.1	Reports Distribution
1	522.1	Unclassified Lib (C) (1 m)
1	522.2	Unclassified Lib (A)
1	93	

DTNSRDC ISSUES THREE TYPES OF REPORTS

1. DTNSRDC REPORTS, A FORMAL SERIES, CONTAIN INFORMATION OF PERMANENT TECHNICAL VALUE. THEY CARRY A CONSECUTIVE NUMERICAL IDENTIFICATION REGARDLESS OF THEIR CLASSIFICATION OR THE ORIGINATING DEPARTMENT.

2. DEPARTMENTAL REPORTS, A SEMIFORMAL SERIES, CONTAIN INFORMATION OF A PRELIMINARY, TEMPORARY, OR PROPRIETARY NATURE OR OF LIMITED INTEREST OR SIGNIFICANCE. THEY CARRY A DEPARTMENTAL ALPHANUMERICAL IDENTIFICATION.

3. TECHNICAL MEMORANDA, AN INFORMAL SERIES, CONTAIN TECHNICAL DOCUMENTATION OF LIMITED USE AND INTEREST. THEY ARE PRIMARILY WORKING PAPERS INTENDED FOR INTERNAL USE. THEY CARRY AN IDENTIFYING NUMBER WHICH INDICATES THEIR TYPE AND THE NUMERICAL CODE OF THE ORIGINATING DEPARTMENT. ANY DISTRIBUTION OUTSIDE DTNSRDC MUST BE APPROVED BY THE HEAD OF THE ORIGINATING DEPARTMENT ON A CASE-BY-CASE BASIS.


DATE
FILMED
7-8